Contenido de la actividad

Fecha de entrega: 26 de noviembre 2021 a las 8:00hs de la mañana.

Metodología de Trabajo:

Realizaremos un trabajo integrador: "DESARROLLO DE PROYECTO".

El mismo formará parte de un trabajo de investigación sobre los temas enfocados a lo largo de la cursada, orientándolos a la elección de un motor de base de datos.

Trabajo práctico Individual.

Se presentará sobre el final del cuatrimestre.

Vínculo con las unidades de trabajo:

Sobre la base de los contenidos trabajados a lo largo de la cursada, se podrán aplicar los temas abordados en las diferentes unidades, con el fin de que formen parte del alcance de investigación del trabajo propuesto.

Por ejemplo: Estructuras, tipos de datos, administración de datos, etc

Actividad: Planteo del Desafío.

Trabajo Práctico de Investigación NoSQL

Dada la experiencia laboral y/o académica adquirida en su entorno de trabajo o bien a lo largo de la cursada, se solicita desarrollar un Trabajo Práctico de Investigación orientado a un motor o al concepto de NoSQL de su elección.

El mismo deberá contar con al menos 6 aspectos de la materia como alcance del trabajo (Introducción, trabajos relacionados, ventajas, desventajas, etc.).

Para ello se le solicita:

Un archivo en formato "pdf" con los siguientes contenidos:

- Portada (con datos principales del alumno/a;
- Índice de temas propuestos;
- Contenidos del tema tratado;
- Conclusión/reflexión del estudiante y cómo se integran los contenidos de la materia;
- Referencias Bibliográficas utilizadas.

Para este trabajo, no se deben superar las 25/30 hojas.

Presentar un archivo pdf con todo el contenido, este incluirá un link a una grabación de la explicación de dicho trabajo (no más de 5 min). Deberá ser un link de oculto en Youtube.

INDICE

Introducción:

Para ponernos en contexto, vamos a empezar hablando sobre base de datos NoSQL, también conocido como “Not only SQL” o “No solamente SQL”.

Las bases de datos no relacionales, están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Son ampliamente reconocidas porque son fáciles de desarrollar, tanto en funcionalidad como en rendimiento a escala.

Las bases de datos no relacionales (NoSQL) son las que, a diferencia de las relacionales, no tienen un identificador que sirva de relación entre un conjunto de datos y otros.

Este tipo de base de datos se dividen en 4, según su utilidad:

- Key Value Stores: Guardan la información en formato de llaves y valores. Se usan para guardar cache, información de sesión de los usuarios o cosas muy sencillas.
- Graph Databases: Bases de datos basadas en Grafos. Nos permiten establecer conexiones entre nuestras entidades para realizar consultas de una forma más eficiente que en bases de datos relacionales.
- Wide-column Stores: Bases de datos columnares. Tienen una llave de fila y otra de columnas para hacer consultas muy rápidas y guardar grandes cantidades de información, pero modelar los datos se puede volver un poco complicado.
- Document Databases: Bases de datos basadas en documentos. Nos permiten guardar documentos dentro de colecciones, tiene muy

buena performance y flexibilidad que nos permite modelar casos de la vida real de forma sencilla y efectiva.

El trabajo de investigación se basa en comprender la Apache CouchDB que se basa en document databases o base de datos basadas en documentos.
La elección del motor CouchDB, se fundamenta en que es una base de datos que abarca completamente su funcionamiento con aplicaciones web y móviles modernas. Almacena sus datos con documentos JSON, utilizando el protocolo HTTP, al realizar las consultas y la devolución de documentos lo hace a través de JavaScript. Pero lo que mas me llamó la atención y fue crucial para la elección, es la historia detrás del proyecto. CouchDB en principios fue un proyecto creado por Damien Katz, un ex desarrollador de IBM. Él mismo financio su proyecto y luego de 2 años lo liberó como proyecto de código abierto. Luego de un año el proyecto paso a manos de la Apache Software Foundation, que es una organización sin fines de lucro, descentralizada de desarrolladores que trabajan con proyectos de código abierto.
De este modo lo que me atrajo a realizar la investigación fueron los principios representan aquellos proyectos de código abierto y sobre todo que su creados fue una persona emprendedora que confío su proyecto y deicidio invertir en él no solo su tiempo, si no también su dinero.

Que es Apache CouchDB?

Apache CouchDB pertenece a una nueva generación de sistemas de gestión de bases de datos.

El modelo de documento sin esquema de CouchDB se adapta mejor a las aplicaciones comunes, cómo el motor de consultas integrado es una forma poderosa de usar y procesar sus datos, y cómo el diseño de CouchDB se presta a la modularización y la escalabilidad.

El diseño de CouchDB se basa en gran medida en la arquitectura web y los conceptos de recursos, métodos y representaciones. Aumenta esto con poderosas formas de consultar, mapear, combinar y filtrar sus datos. Agrega tolerancia a fallas, escalabilidad extrema y replicación incremental. Cuando hablamos de mapear estas haciendo referencias a vistas.

Las vistas son útiles para muchos propósitos:

- Filtrar los documentos en su base de datos para encontrar aquellos relevantes para un proceso en particular.
- Extraer datos de sus documentos y presentarlos en un orden específico.
- Construyendo índices eficientes para encontrar documentos por cualquier valor o estructura que resida en ellos.
- Utilice estos índices para representar relaciones entre documentos.
- Finalmente, con las vistas puede realizar todo tipo de cálculos sobre los datos de sus documentos. Por ejemplo, si los documentos representan las transacciones financieras de su empresa, una vista puede responder a la pregunta de cuál fue el gasto en la última semana, mes o año.

Estructura

CouchDB aloja bases de datos con nombre, que almacenan documentos. Cada documento tiene un nombre único en la base de datos, y CouchDB proporciona una API HTTP RESTful para leer y actualizar (agregar, editar, eliminar) documentos de la base de datos.

Los documentos son la unidad principal de datos en CouchDB y constan de cualquier número de campos y archivos adjuntos. Los documentos también incluyen metadatos que mantiene el sistema de base de datos. Los campos del documento tienen un nombre único y contienen valores de diferentes tipos (texto, número, booleano, listas, etc.), y no hay un límite establecido para el tamaño del texto o el recuento de elementos.

Las ediciones de documentos se realizan mediante aplicaciones cliente que cargan documentos, aplican cambios y los guardan en la base de datos. Si otro cliente que edita el mismo documento guarda sus cambios primero, el cliente obtiene un error de conflicto de edición al guardar. Para resolver el conflicto de actualización, se puede abrir la última versión del documento, volver a aplicar las ediciones y volver a intentar la actualización.

Las actualizaciones de un solo documento (agregar, editar, eliminar) son todo o nada, ya sea exitosamente o fallando completamente. La base de datos nunca contiene documentos parcialmente guardados o editados.

Para proteger quién puede leer y actualizar documentos, CouchDB tiene un modelo de validación de actualización y acceso de lector simple que se puede ampliar para implementar modelos de seguridad personalizados.

Principales características.

CouchDB es un sistema de base de datos distribuido basado en pares. Permite a los usuarios y servidores acceder y actualizar los mismos datos compartidos mientras están desconectados. Posteriormente, esos cambios se pueden replicar bidireccionalmente.

Los modelos de seguridad, visualización y almacenamiento de documentos de CouchDB están diseñados para trabajar juntos para hacer que la replicación bidireccional sea eficiente y confiable. Tanto los documentos como los diseños se pueden replicar, lo que permite que las aplicaciones de bases de datos completas (incluido el diseño de la aplicación, la lógica y los datos) se repliquen en computadoras portátiles para su uso sin conexión o en servidores en oficinas remotas donde las conexiones lentas o poco confiables dificultan el intercambio de datos.

El proceso de replicación es incremental. En el nivel de la base de datos, la replicación solo examina los documentos actualizados desde la última replicación. Si la replicación falla en cualquier paso, debido a problemas de red o fallas, por ejemplo, la siguiente replicación se reinicia en el último punto de control.

Se pueden crear y mantener réplicas parciales. La replicación se puede filtrar mediante una función de JavaScript, de modo que solo se replican documentos particulares o aquellos que cumplen criterios específicos. Esto puede permitir a los usuarios desconectar subconjuntos de una gran aplicación de base de datos compartida para su propio uso, mientras mantienen una interacción normal con la aplicación y ese subconjunto de datos.

Ofrecen un rendimiento mayor y sus modelos de datos resuelven varios problemas que no se plantearon al definir el modelo relacional como, por ejemplo:

- Grandes volúmenes de datos estructurados, semi-estructurados y sin estructurar. Casi todas las implementaciones NoSQL ofrecen algún tipo de representación para datos sin esquema, lo que permite comenzar con una estructura y con el paso del tiempo, añadir nuevos campos, ya sean sencillos o anidados a datos ya existentes.

- Sprints ágiles, iteraciones rápidas y frecuentes commits/pushes de código, al emplear una sintaxis sencilla para la realización de consultas y la posibilidad de tener un modelo que vaya creciendo al mismo ritmo que el desarrollo.

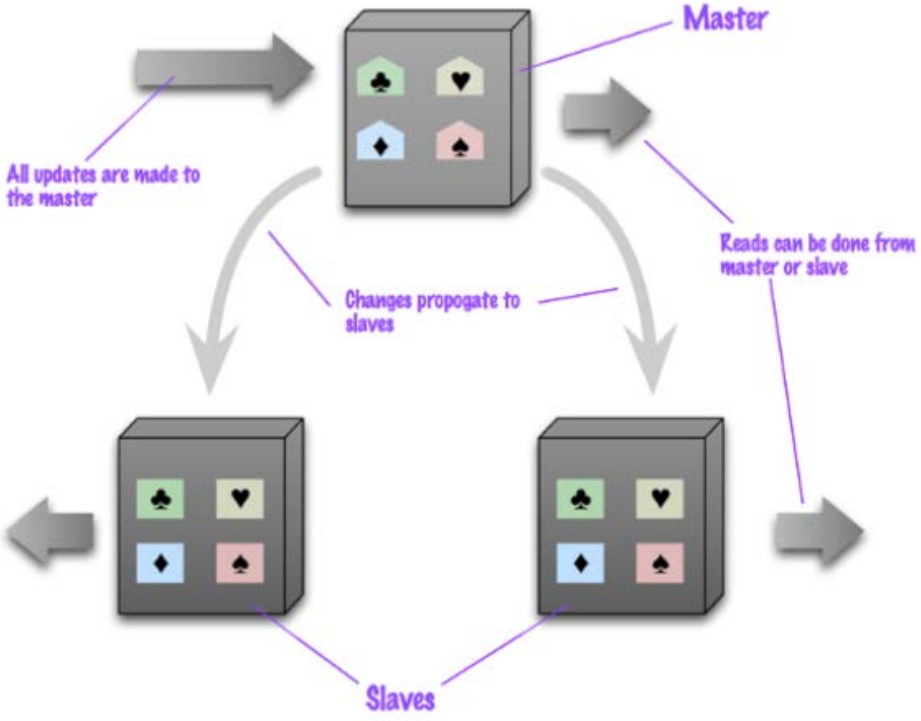
- Arquitectura eficiente y escalable diseñada para trabajar con clusters en vez de una arquitectura monolítica y costosa.

Replicaciones en CouchDB

Comenzaré comentado que son las replicaciones en NoSQL y luego desarrollaremos como son abordadas en Apache CouchBD.

La replicación mantiene copias idénticas de los datos en múltiples servidores, lo que facilita que las aplicaciones siempre funcionen y los datos se mantengan seguros, incluso si alguno de los servidores sufre algún problema.

La mayoría de las bases de datos NoSQL también soportan la replicación automática, lo que implica una alta disponibilidad y recuperación frente a desastres sin la necesidad de aplicaciones de terceros encargadas de ello. Desde el punto de vista del desarrollador, el entorno de almacenamiento es virtual y ajeno al código de aplicación.

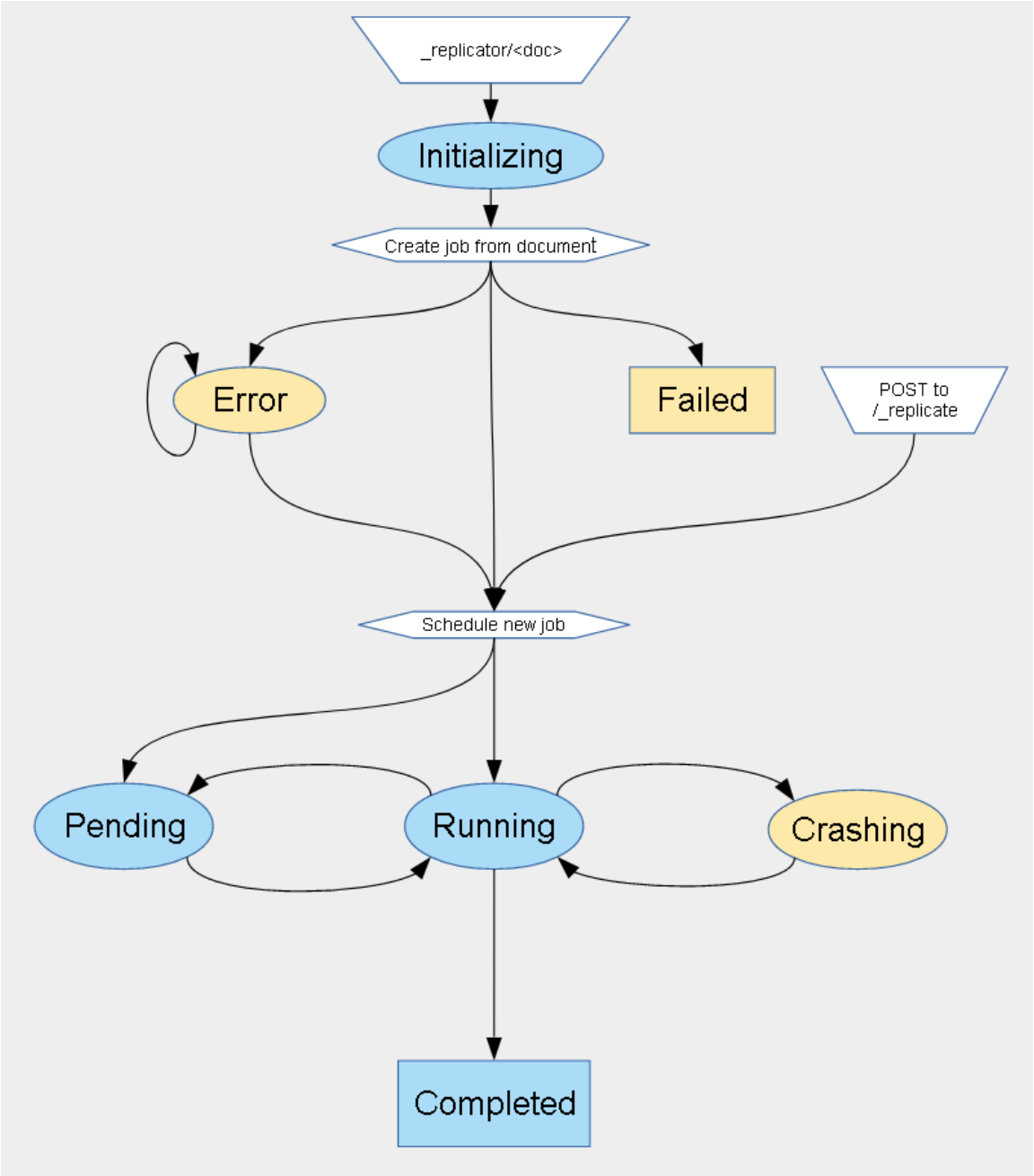


La replicación de CouchDB es uno de estos componentes básicos. Su función fundamental es sincronizar dos o más bases de datos.Para permitir que la replicación resuelva una serie de problemas: sincronice de manera confiable las bases de datos entre múltiples máquinas para el almacenamiento de datos redundantes; distribuir datos a un clúster de instancias que comparten un subconjunto del número total de solicitudes que llegan al clúster (equilibrio de carga); y distribuir datos entre ubicaciones físicamente distantes.

La replicación de CouchDB usa la misma API REST que usan todos los clientes. La replicación funciona de forma incremental; es decir, si durante la replicación algo sale mal, como interrumpir la conexión de red, la próxima vez que se ejecute continuará donde lo dejó. Además, solo transfiere los datos necesarios para sincronizar bases de datos.

Estados de replicación

Los trabajos de replicación durante su ciclo de vida pasan por varios estados. Este es un diagrama de todos los estados y transiciones entre ellos:

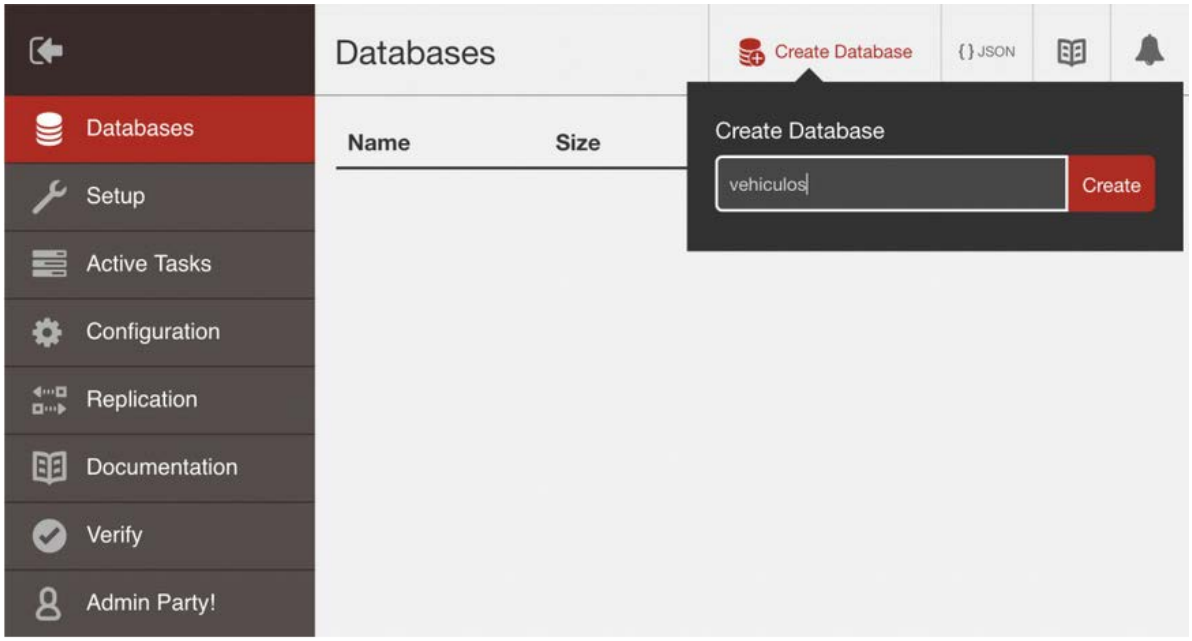

_replicator/<doc>
Initializing
Create job from document
Error
Failed
POST to
/_replicate
Schedule new job
Pending
Running
Crashing
Completed

Funcionamiento e Interfaz

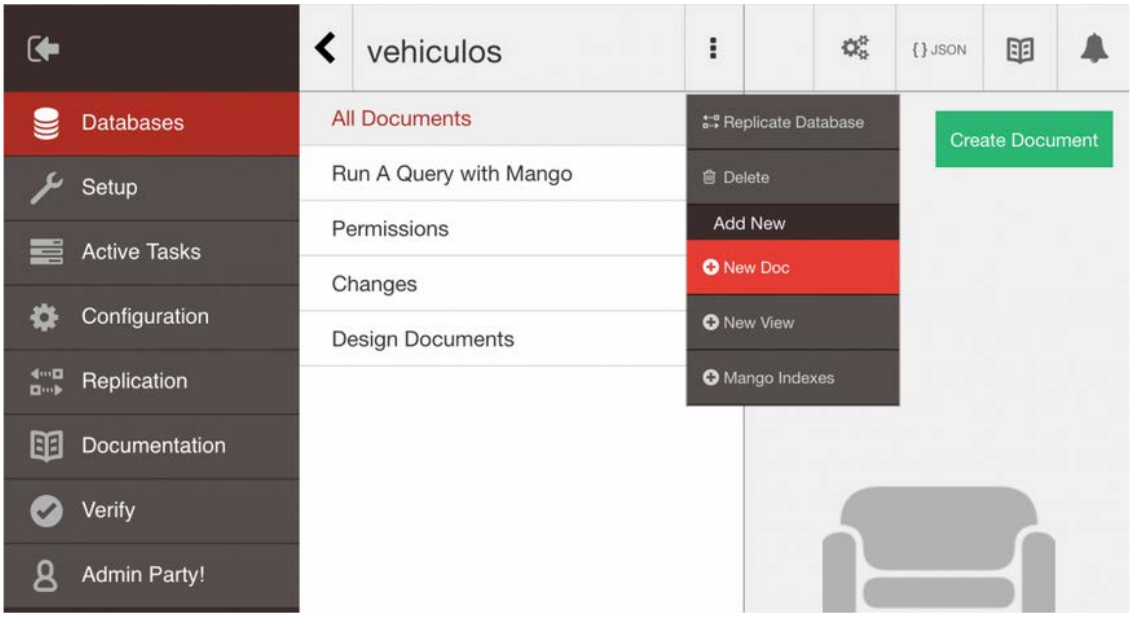
Desarrollaremos una serie de comparaciones, acompañadas de capturas de CouchBD, con nuestra forma de trabajar en base de datos relacionales.

Tener en cuenta que a diferencia del software que veníamos trabajando como el SQL Server, para ingresar al CouchDB, tenemos que ingresar por localhost (127.0.0.1) en el puerto 5984 desde el navegador.

- Creación de una base de datos:



- Luego se crea un nuevo documento, pinchando puntos suspensivos en nuestra base de datos de vehículos. En base de datos relacionales crearíamos tablas.

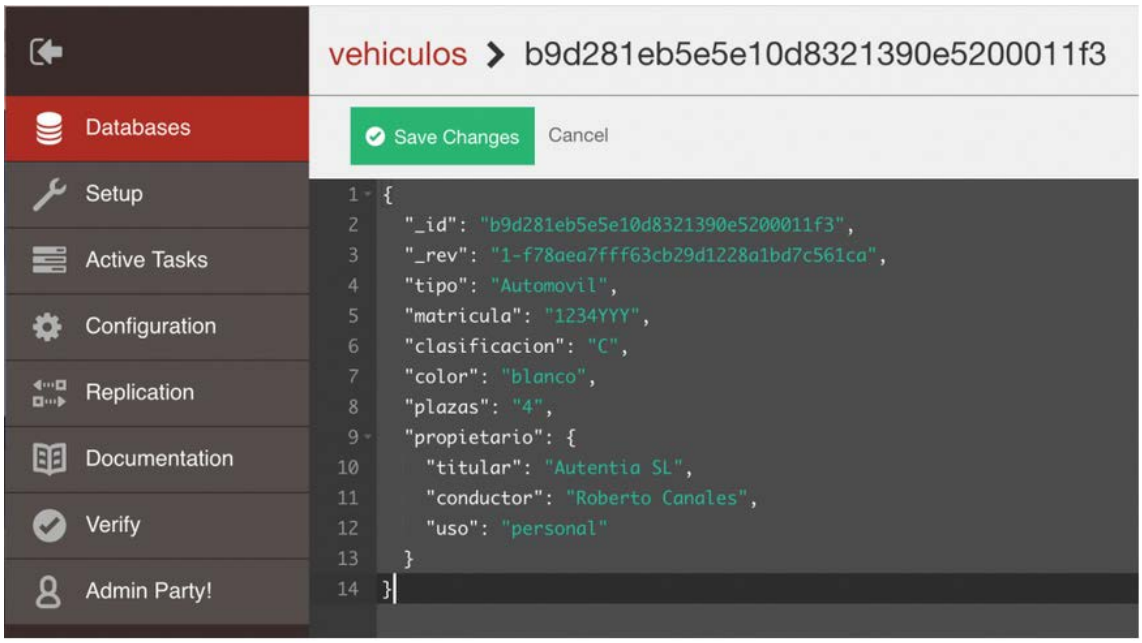


Podemos ver que son documentos JSON, y tiene un identificador _id, que en SQL Server vimos como Primary Key, creando las relaciones con las Foreign Key

Shell

```
{
  "_id": "b9d281eb5e5e10d8321390e520005d21",
    "tipo": "Motocicleta",
  "matricula": "333YYZ",
  "clasificacion": "B",
  "color": "Negro",
  "plazas": "2"
}

{
  "_id": "b9d281eb5e5e10d8321390e520006193",
  "tipo": "Tractor",
  "matricula": "E99999",
  "clasificacion": "E",
  "color": "azul",
  "plazas": "2",
  "carga_maxima": "20T"
}
```

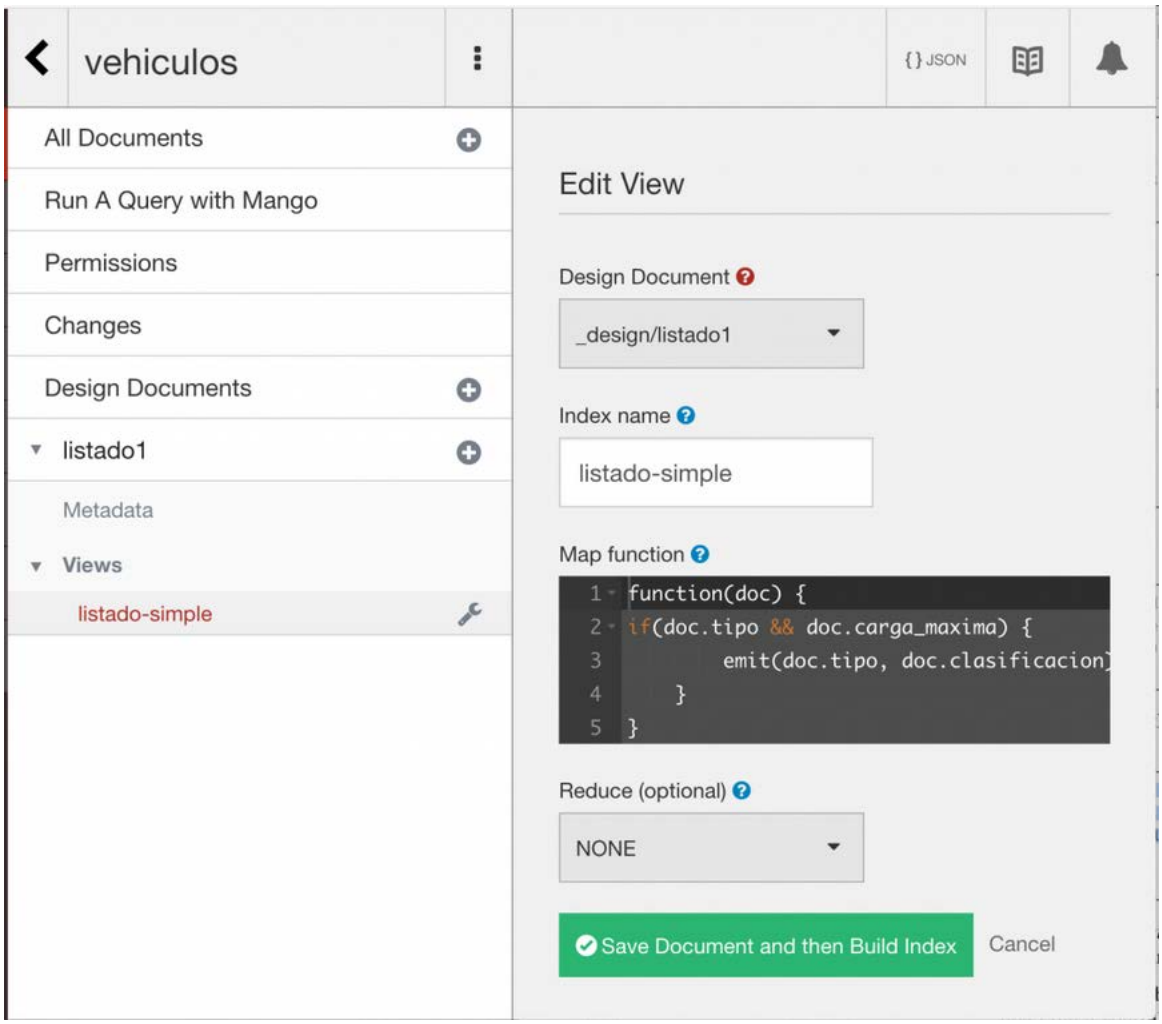


- Vamos a crear una vista, es decir un script para organizar los grupos que devuelven un llamado.

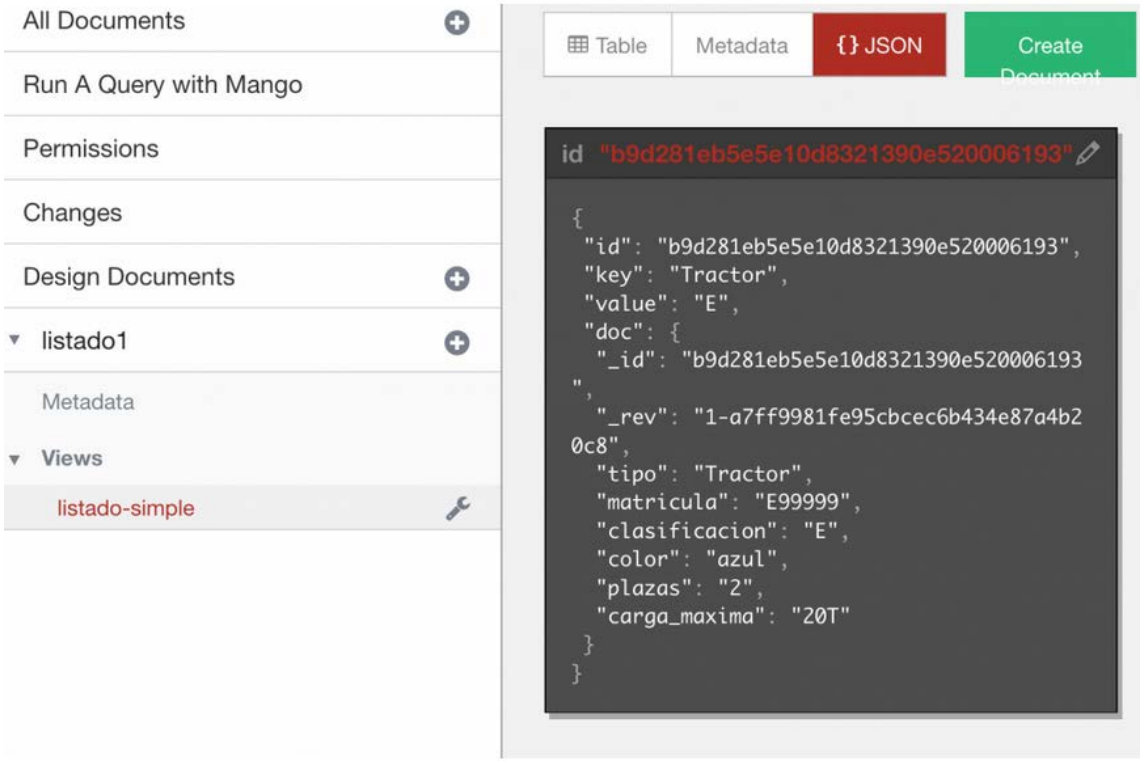
  En este caso, quiero consultar los vehículos que tienen carga máxima en los campos.

  Es por eso que creamos una vista llamada listado-simple, teniendo en cuenta que la dinámica que las bases de datos NoSQL es distinta y utiliza el concepto de funciones map-reduce para recuperar información.

  Siguiente a esto, creamos una función por la que pasan todos los documentos, verificando que los vehículos tienen tipo y carga_maxima, es decir, si tienen los campos que se interesan y todos los demás son descartados.


vehiculos
{} JSON
All Documents
Run A Query with Mango
Permissions
Changes
Design Documents
listado1
Metadata
Views
listado-simple
Edit View
Design Document
_design/listado1
Index name
listado-simple
Map function
function(doc) {
if(doc.tipo && doc.carga_maxima) {
emit(doc.tipo, doc.clasificacion
}
}
Reduce (optional)
NONE
Save Document and then Build Index
Cancel

Podemos ver el resultado como tabla de documentos devueltos, metadatos (la respuesta) o el documento JSON.



- CouchBD también nos permite crear respuestas automáticas de visualizaciones creando vistas tipo show.

Beneficios y Desventajas

Escalabilidad

El diseño arquitectónico de CouchDB lo hace extremadamente adaptable al particionar bases de datos y escalar datos en múltiples nodos. CouchDB admite tanto la partición horizontal como la replicación para crear una solución fácilmente administrada para equilibrar las cargas de lectura y escritura durante la implementación de una base de datos.

Sin bloqueos de lectura

En la mayoría de las bases de datos relacionales, donde los datos se almacenan en tablas, si alguna vez necesita actualizar o modificar una tabla, la fila de datos que se cambia queda bloqueada para otros usuarios hasta que se procesa la solicitud de modificación. Esto puede crear problemas de accesibilidad.

CouchDB usa MVCC (Control de concurrencia de múltiples versiones) para administrar el acceso a las bases de datos al mismo tiempo. Esto significa que, independientemente de las cargas actuales de la base de datos, puede ejecutarse y sin restricciones.

Desarrollo de código abierto

Debido a su fuerte respaldo y soporte en la comunidad de código abierto, CouchDB mantiene una base sólida y confiable para la administración de bases de datos empresariales.

Soporte

Las bases de datos NoSQL al ser de código abierto poseen un soporte diferente al soporte que ofrecen las compañías comerciales a sus productos. Esto puede presentar algunas ventajas y también algunas desventajas.

Falta de experiencia

No hay una gran cantidad de desarrolladores y administradores que conocen la tecnología, lo que hace difícil a las empresas encontrar personas con los conocimientos técnicos apropiados.

Problemas de compatibilidad

A diferencia de las bases de datos relacionales, que comparten ciertos estándares, las bases de datos NoSQL, como CouchDB, tiene su propia API, esto hace que las interfaces de consultas son únicas y tienen peculiaridades.

CONCLUSIÓN

Facilidad al usar esta base de datos NoSQL hace que los procedimientos parezcan simples y sencillos, creo que esta le da una ventaja sobre los demás gestores de bases de datos.

CouchDB se adapta perfectamente a la web con su interfaz y de manera amigable se puede realizar operaciones, replicas entre otros y muestra tolerancia a los fallos.

Aunque específicamente sobre CouchDB, como base de datos No relacional no hay muchos tutoriales/videos, mostrando su uso y detalles a posibles errores que puedan surgir en su implementación.

Video: https://youtu.be/pEDYgFwjrVY

Bibliografía:

https://docs.couchdb.org/en/stable/

http://expertojava.ua.es/si/nosql/nosql.html#sesion01

https://www.ibm.com/cloud/learn/couchdb

https://es.wikipedia.org/wiki/CouchDB#Caracter%C3%ADsticas_principales

https://www.adictosaltrabajo.com/2018/11/27/primeros-pasos-con-couchdb/